End-User Probabilistic Programming

End-User Probabilistic Programming

Judith Borghouts, Andrew D. Gordon, Advait Sarkar, and Neil Toronto

Microsoft Research

Abstract. Probabilistic programming aims to help users make decisions under uncertainty. The user writes code representing a probabilistic model, and receives outcomes as distributions or summary statistics. We consider probabilistic programming for end-users, in particular spreadsheet users, estimated to number in tens to hundreds of millions. We examine the sources of uncertainty actually encountered by spreadsheet users, and their coping mechanisms, via an interview study. We examine spreadsheet-based interfaces and technology to help reason under uncertainty, via probabilistic and other means. We show how uncertain values can propagate uncertainty through spreadsheets, and how sheet-defined functions can be applied to handle uncertainty. Hence, we draw conclusions about the promise and limitations of probabilistic programming for end-users.

1 Introduction

In this paper, we discuss the potential of bringing together two rather distinct approaches to decision making under uncertainty: spreadsheets and probabilistic programming. We start by introducing these two approaches.

1.1 Background: Spreadsheets and End-User Programming

The spreadsheet is the first "killer app" of the personal computer era, starting in 1979 with Dan Bricklin and Bob Frankston's VisiCalc for the Apple II [15]. The primary interface of a spreadsheet—then and now, four decades later—is the *grid*, a two-dimensional array of cells. Each cell may hold a literal data value, or a formula that computes a data value (and may depend on the data in other cells, which may themselves be computed by formulas). Spreadsheets help democratise computing by allowing computer users to create their own customised calculations based on their own data. They are highly flexible and general-purpose, capable of performing a huge variety of jobs for a great many users in their working or personal lives.

Spreadsheet formulas comprise calls to a wide collection of built-in algorithms, encapsulated in functions known as *worksheet functions*. Formulas typically act on strings, numbers, two-dimensional arrays, and can treat fragments of the grid as arrays. Formulas may consist of complex, nested expressions, including conditionals and other forms of control flow. For these and other reasons, spreadsheets can be viewed as code [16], and spreadsheet users are canonical

examples of *end-user programmers* [18]: people who write code primarily for their own use. Even though they write code, end-user programmers are usually not professional developers. An end-user programmer often has little intrinsic interest or education in computing but instead wishes to get some job done with the spreadsheet. They are "business professionals or scientists or other kinds of domain specialists whose jobs involve computational tasks" [24].

Spreadsheets often contain uncertain data: for example, academics may deal with noise and missing data in their data sets, managers may have to make business decisions based on projected sales data, and project leaders have to adapt schedules based on estimated workload. Some of the core affordances of spreadsheets are mechanisms to deal with uncertainty: for instance, uncertainty about future events can be modelled simply by trying out different parameters and immediately seeing an updated model. Due to their flexibility, ubiquity, and low knowledge barriers, spreadsheets are acknowledged to be a "breakthrough technology for practical modeling" [28]. Still, this paper considers some proposed additions to spreadsheets to propagate uncertain values through calculations and models.

1.2 Background: Probabilistic Programming

Let's turn to another approach to decision making under uncertainty: statistical models. The purpose of a statistical model is to infer insights from observed data. Much expertise is needed to write, and interpret the results of, statistical inference algorithms, such as randomised Monte Carlo methods or deterministic message-passing. The aim of *probabilistic programming* [12] is to empower domain experts and expert statisticians to get the benefits of statistical modelling and machine learning, without needing expertise in writing inference algorithms. The idea is that the user specifies a statistical model by writing a piece of code, and delegates the difficulty of statistical inference to an automatic compiler.

Probabilistic programming languages typically comprise a deterministic core language, plus (1) operations to sample from probability distributions, (2) operations to condition on observations, and (3) operations to infer properties of the resulting probability distributions. BUGS [9] is the first probabilistic programming language, first developed in 1989 [14], and used extensively in several textbooks for statisticians and social scientists [8,25,20]. Infer.NET [23], developed since 2004, is used at scale in Microsoft. BUGS and Infer.NET only support certain classes of graphical models. Church [10] introduced the idea of a *universal* probabilistic programming language, that can express any model written in a Turing-complete programming language (although efficient inference in general remains a challenge). More recently, Stan [6] and PyMC3 [21] have also gained wide popularity, and there is a wide range of research languages, including Figaro [27], Anglican [37], and many others. Probabilistic programming environments with graphical representations have also been developed, to aid the understanding of programmers new to the paradigm [13].

programming (writing formulas) in spreadsheets
(tens to hundreds of millions of people)
>>>
probabilistic programming (without conditioning) in spreadsheets
(hundreds of thousands of people)
>
probabilistic programming in probabilistic programming languages
(tens of thousands of people)

(Published estimates of spreadsheet users range from tens [32] to hundreds of millions `https://irishtechnews.ie/seven-reasons-why-excel-is-still-used-by-half-a-billion-people-worldwide/`. Palisade, the maker of @Risk, claims use by over 150,000 decision makers. See `https://www.palisade.com/about/about.asp`. RStan has about 20K downloads per month and the Stan website has about 15K unique visitors per month (personal communication, Matthijs Vákár, May 2019). See also `https://discourse.mc-stan.org/t/estimating-popularity-of-stan-and-related-packages/8768`.)

Table 1: Estimated users of probabilistic programming and of spreadsheets.

1.3 Bringing Probabilistic Programming to the Spreadsheet

Why might we want to enable probabilistic programming in spreadsheets? As we have already discussed, a substantial amount of decision making around the world is supported by data in spreadsheets. Many models of uncertain situations such as financial plans, events, scientific experiments, and so on, are built using spreadsheet formulas by end-user programmers.

Thus, the direction seems inevitable: let's take probabilistic programming to the data, to the spreadsheet! These observations have led researchers on probabilistic programming languages (including one of the authors) to design probabilistic programming systems aimed towards spreadsheet users. Examples include Tabular [11] and Invrea's Scenarios [39].

In fact, probabilistic modelling and even aspects of probabilistic programming have existed in spreadsheets from early on, before the interest in probabilistic programming for statistics and machine learning.

The formula RAND() draws at random from the uniform distribution on the unit interval. The formula NORM.INV(RAND(),0,1) draws at random from the standard normal distribution. Writing Monte Carlo simulations using such randomized spreadsheet formulas is a simple form of probabilistic modelling, based on repeated sampling. Monte Carlo simulations can be implemented, for example, by arranging a randomised computation in a row of a sheet, and then replicating the row many times.[1] Books on spreadsheet modelling devote whole chapters to this idiom [28,38]. Savage [31] advocates probabilistic modelling using features such as Excel's data tables.[2]

[1] See `https://www.youtube.com/watch?v=BQv2Uyea8i4&t=27s`, for example.
[2] See `https://www.probabilitymanagement.org/`.

A key aspect of probabilistic programming is that the user writes a model, and the system handles inference. Writing Monte Carlo simulations by replicating some formulas is not probabilistic programming as the user is expressing the inference algorithm directly. Still, there are well-established add-ins that do support probabilistic programming without tedious replication by the user. These include @Risk (pronounced 'at risk', first released in 1987) or Crystal Ball. It seems to be uncommon, but these tools also support forms of Bayesian conditioning via rejection sampling. (Probabilistic programming languages support conditioning via inference techniques that are far more efficient than rejection sampling.)

Intriguingly, we can reason that there are more users of probabilistic programming in spreadsheets via these add-ins than in actual probabilistic programming languages such as BUGS or Stan. Table 1 shows rough estimates of orders of magnitude of users today. We cannot be certain, of course, because we have only rough estimates of usage numbers. End-user probabilistic programming does appear to be a relatively miniscule subset of all end-user programming: the use of formulas for probabilistic modelling is probably a tiny fraction of the use of formulas in general.

1.4 How Would Probabilistic Programming Help Spreadsheet Users?

To understand how better support for probabilistic programming might help end users, we conducted an interview study of how spreadsheet users manage uncertainty. The study used thematic analysis [5], a qualitative method, common in psychology, in which transcripts of the interviews are *coded* (that is, labelled by researchers) to mark significant phenomena, and the results aggregated.

This paper reports some technical background, the interview study itself, and the design implications of the study.

We begin in Section 2 by describing two different existing proposals for spreadsheet extensions that can deal with uncertainty. First, we describe *uncertain values*, that can be used like ordinary certain values, and that propagate uncertainty through calculations. We describe three formalisms for uncertain values: qualitative, possibilistic, and probabilistic. Second, we describe *sheet-defined functions*, and how they can be applied to model uncertainty. Section 3 describes our interview study and its findings, including a categorisation of the types of uncertainty encountered by spreadsheet users in spreadsheets they had constructed, and also a categorisation of the coping strategies adopted by the spreadsheet users. Section 4 describes design implications of the interview study, and explores how the formalisms of Section 2 apply to the categorisations of uncertainty faced by users in Section 3.

To the best of our knowledge, this work is the first study of how end-users deal with uncertain values in spreadsheets, and the first to discuss how various candidate spreadsheet extensions might match the potential needs of end-users. We augment Streit's framework with the idea of using arrays of possible scenarios or probabilistic samples as a representation of uncertain values.

2 Spreadsheet Extensions for Uncertainty

We consider how spreadsheets can be extended to better handle different types of uncertainty. We consider two extensions. First, Streit [35] proposed to store different sorts of uncertain value in cells, and have these uncertain values propagate through calculations. Second, Peyton Jones, Blackwell, and Burnett [26] proposed sheet-defined functions as a general-purpose mechanism for end-users to define new worksheet functions by an example calculation in a sheet. We explain how both these mechanisms can help users manage uncertainty. We do so with a running example, which we present next.

2.1 Example: Clara's Budget

Let us consider Clara,[3] a fictitious character who represents a common category of spreadsheet end-user. She is a confident computer user. She does not identify as a programmer, but does use spreadsheets in her work and personal life. She is confident to model with simple formulas including arithmetic and common functions like SUM, but is not highly confident with more complex formulas.

We join Clara as she is preparing a spreadsheet to help decide whether she should purchase a sofa, given her budget for this month. The equations below show her assignments of data and formulas to cells.

```
E1 = "Budget for Jan 2018"
E3 = "Income"; F4 = 2000
E5 = "Expenses"
E6 = "Rent"; F6 = 1100  // A certain cost
E7 = "Commute"; F7 = 85 // Another certain cost
E8 = "Sofa!"; F8 = 700  // Clara is deciding whether to buy this item
E9 = "Utilities"; F9 = 100  // But she is uncertain about her utilities bill
E10 = "Total expenses"; F10 = SUM(F6:F9) // value: 1985
E12 = "Balance"; F12 = F4−F10 // value: 15
```

(Instead of the standard view of the spreadsheet, which shows the values of formulas but not the formulas themselves, we use a textual notation for spreadsheets known as Calculation View [29], and show values of formulas in comments.)

Clara lists out her certain costs, rent and commute, the cost of the sofa, and puts in her estimate of the utilities bill. She calculates in F12 the balance of her income given her total expenses, calculated by the formula in F10.

Clara wants the sofa but doesn't want her total expenses to exceed her income. She is uncertain about her utilities bill. How can the spreadsheet help her decide what to do?

[3] To be clear, Clara is a fictional character, contrived to illustrate these technical solutions to representing uncertainty. Our interview study looked at how actual users deal with uncertainty in today's spreadsheets. It would be interesting in future to get the reactions of real people to the technical solutions presented in this section.

2.2 Managing Uncertainty with Uncertain Values

In this section, we turn to a technical approach to handling uncertainty in spreadsheets based on storing *uncertain values* in cells. Most spreadsheet systems allow a cell to hold only an individual value, such as a text or a number. Streit [35] proposed and implemented a spreadsheet where a cell can hold an value of an *uncertainty type*, such as a number explicitly tagged as an estimate, or a numeric interval like 7 ± 2, or a probability distribution such as a normal distribution with parameters for mean and standard deviation.

Other researchers have proposed aggregate values in cells, such as arrays [2], which could in principle represent uncertain values, but to the best of our knowledge Streit was the first to consider uncertain values explicitly.

Streit proposed to enrich the spreadsheet interface in several ways:

1. The user can input uncertainty information into cells. Input can be via a textual notation, such as a numeric range. Input could also be assisted by some interface that gathers parameters of a probability distribution, for instance.
2. Uncertainty information propagates through formula evaluation.
3. The presence of uncertainty information is indicated within cells. A most likely value might be displayed, for example, together with an indication that the value is uncertain.

Unlike Streit, in the following classification we do not consider visualization techniques. Visualization of uncertainty [36] is an important subject, but outside the scope of this paper. In terms of semantics, uncertain values and their propagation through formula evaluation are a kind of computational effect [1]. To the best of our knowledge, there has been no formal semantics for Streit's uncertain values. That too would be an interesting challenge for future research.

Next, we discuss three kinds of uncertain value—Qualitative, Possibilistic, and Probabilistic—that could be implemented in spreadsheet systems.

Qualitative The simplest form of uncertain value is an *estimate*, a qualitative indication that the value is approximate.

- The function ESTIMATE(V) returns the value V tagged as uncertain.

Clara can indicate that she is uncertain about her utilities bill as follows:

E9 = `"Utilities"`; F9 = ESTIMATE(100)

(In practice, Clara would likely use some graphical interface to help enter the ESTIMATE tag, but we omit the details.)

Estimates propagate through calculation, so that the formulas depending on cell F9 also return estimates.[4]

```
F10 = SUM(F6:F9)  // value: ESTIMATE(1985)
F12 = F4−F10  // value: ESTIMATE(15)
```

[4] Readers might notice a parallel to the literature on information flow and 'tainting'.

Clara's simple spreadsheet gains little from this qualitative tracking of uncertainty. Still, larger spreadsheets with many values, some uncertain, some not, may benefit more as there would be a clear indication of which results are certain, in spite of the presence of uncertain values.

The implementation of qualitative uncertainty as the function ESTIMATE(V) is merely one possibility. One might also imagine an alternative implementation, invoked through buttons/menus, where the 'tag' is applied as cell metadata, much like cell formatting. While easier to access for non-expert end-users, this implementation would also be less flexible; it would not be possible to mark values within larger expressions as being uncertain, for example.

Possibilistic There are several possibilistic approaches to uncertainty, where an uncertain value represents a set of possible values. In one, the *multiple scenarios* approach, the uncertain value consists of a tuple of values, each corresponding to one of N scenarios. For example, $N = 3$ could represent best case, most-likely case, and worst case.

- The function SCENARIOS(V1,...,VN) returns an uncertain value representing N scenarios.

Suppose Clara considers the best case cost of her utilities to be 50, the most-likely 100, and the worst case to be 150. She can write this as follows:

E9 = `"Utilities"`; F9 = SCENARIOS(50,100,150)

These scenarios propagate through her calculations like this:

F10 = SUM(F6:F9) // value: SCENARIOS(1935,1985,2035)
F12 = F4−F10 // value: SCENARIOS(65,15,−35)

The spreadsheet tells Clara that, given her assumptions, her best case balance is 65, most-likely 15, and worst case -15.

Having a multiple-scenario value propagate through the sheet is equivalent to Clara trying out the three input values one at a time: the classic 'what-if' analysis afforded by spreadsheets. Still, the advantage over one at a time entry is that the spreadsheet can display the three possibilities side-by-side in the cell. Also, if the other costs are uncertain, Clara can enter those too as multiple-scenarios. One implementation of uncertainty propagation could automatically combine all the 1st cases, all the 2nd cases and all the 3rd cases (which in this simple example corresponds to best-case, most-likely case, and the worst-case), which is easier than for Clara to manually manage the correspondences.[5]

An alternative to these *dependent* scenarios, where an elementwise product of scenarios is taken, is *independent* scenarios (which we have not discussed), where the cartesian product of the individual scenarios would be taken.

[5] Some versions of Microsoft Excel provide a specialist tool, the Scenario Manager, that evaluates multiple copies of a template sheet, one copy for each scenario, and produces a report. The Scenario Manager is a wizard, outside the formula language. Its results are not subject to automatic recalculation. See `https://www.youtube.com/watch?v=c0tdVlPvFZ4`.

Another form of possibilistic uncertainty, considered by Streit and others, is a numeric interval, where propagation consists of interval arithmetic [17].

Probabilistic A possibilistic uncertain value consists of a set of possible values (such as a finite set of scenarios or an infinite set of real numbers in a numeric interval). Additionally, a *probabilistic* uncertain value is a probability distribution: roughly a possibilistic value, a set, together with a weight for each value.

A simple way to introduce weighted estimates is the *triangular distribution*:

- The function DIST.TRIANG(a, b, c) constructs a value distributed according to a *triangular distribution*, with lower bound a, upper bound b, and mode c, where $a < b$ and $a \leq c \leq b$.

The triangular distribution can capture subjective judgments of probabilities, where c is someone such as Clara's best guess, and a and b are subjectively the intuitive minimum and maximum. The shape of the density of this continuous distribution is a triangle that peaks at c and falls to zero at a and b.

If Clara enters a probabilistic approximate value using this function it propagates as follows.

```
E9 = "Utilities"; F9 = DIST.TRIANG(50,150,100)
F10 = SUM(F6:F9)  // value: DIST.TRIANG(1935,2035,1985)
F12 = F4−F10  // value: DIST.TRIANG(−35,65,15)
```

In the case above, the distribution can be propagated exactly. Streit only considered propagation of parametric probability distributions (such as the families of normal distributions or triangular distributions) that can be calculated exactly. We cannot always compute propagated distributions exactly (for example, the sum of two triangular distributions is not a triangular distribution, and in general it may not be possible to derive a closed-form expression for a distribution generated through any arbitrary arithmetic on other probability distributions). Instead approximate representations based on Monte Carlo sampling can be used. A general but approximate representation of a probabilistic uncertain value is an array of samples from the distribution. Such an array is known as a *stochastic information packet* (SIP) [30].[6] A key advantage of SIPs is that it is easy to compute with them; for example, the sum of two triangular distributions represented as SIPs, can be computed (and thereby represented) as the elementwise addition of the individual SIPs.

How does the spreadsheet visualise a probabilistic uncertain value in a cell? A simple option is to show "∼15", that is, the mean 15 together with the sign "∼" to indicate the uncertainty. A more sophisticated representation is to visualize the density, perhaps as a histogram, although it takes practice for end users to make sense of a density.

In Clara's case, she may be most interested in the risk that her balance goes below zero. If she writes the Boolean formula F12<0 to test this event,

[6] Use of SIPs to represent probabilistic values is equivalent to the mechanisms used by @Risk and Crystal Ball. Guesstimate is a spreadsheet-like system that uses SIPs to represent uncertain values. See `https://www.getguesstimate.com/`.

propagation of the triangular distribution in F12 yields the probability that the formula is true.

```
E13="Chance of overdraft"; F13=F12<0 // value: 25%
```

So with probabilistic assumptions, the spreadsheet can tell Clara the probability of modelled events such as her overdraft. In this case, she really wants that sofa, and will take the risk!

2.3 Managing Uncertainty with Sheet-Defined Functions

Observe that when considering multiple chosen scenarios in the possibilistic case, or when considering many randomly drawn scenarios in the probabilistic case, we are replaying the original spreadsheet model with different parameters.

A user can replay a calculation by copying part of a sheet, and changing parameter values. Still, this practice is prone to error as the model gets larger, or if it and its copies need to be updated.

Peyton Jones, Blackwell, and Burnett [26] proposed *sheet-defined functions* as an automatic general alternative to manual replication. It is simply the pervasive idea of procedural abstraction from programming languages, but applied to spreadsheets. A sheet-defined function f is specified by a *body*, a piece of grid, and has a number of input parameters identified by ranges in the body, and an output, also identified by a range in the body. A formula $f(V_1, \ldots, V_n)$ is computed by making a copy of the body, pasting the values V_1, ..., V_n into the parameter ranges, evaluating the body, and then returning the value in the result range.

Sestoft's monograph [**?**] describes how to implement sheet-defined functions. He also describes example usages of sheet-defined functions, including idioms for both possibilistic and probabilistic uncertainty.

To explain these idioms, we first turn Clara's budget into a sheet-defined function as follows. The function Budget has a single parameter, the uncertain utilities bill, held in range F9, and it returns the balance together with a numeric indicator of whether the balance is overdrawn in the range F12:H12.

```
function BUDGET( F9 ) returns F12:G12 {
  E1 = "Budget for Jan 2018"
  E3 = "Income"; F4 = 2000
  E5 = "Expenses"
  E6 = "Rent"; F6 = 1100  // A certain cost
  E7 = "Commute"; F7 = 85 // Another certain cost
  E8 = "Sofa!"; F8 = 700  // Clara is deciding whether to buy this item
  E9 = "Utilities"; F9 = 100  // This cell is the parameter to the SDF
  E10 = "Total expenses"; F10 = SUM(F6:F9)
  E12 = "Balance"; F12 = F4−F10; G12 = IF(F12<0,1,0) // Result cells
}
```

(Here we rely on an extension [22] of Calculation View [29] for a textual notation for sheet-defined functions. Instead, an implementation of sheet-defined functions

would provide a graphical interface for the user to specify the function name, parameters, and other metadata.)

Clara can calculate her three scenarios of possibilistic reasoning with the formulas below. She can write the formula first in C54, and then drag fill to C55 and C56.

```
B54=50;  C54 = BUDGET(B54)  // value: {65,0}
B55=100; C55 = BUDGET(B55)  // value: {15,0}
B56=150; C56 = BUDGET(B56)  // value: {35,1}
```

In some recent spreadsheet systems, an array held in a cell such as C54 *spills* for display into adjacent cells.[7] In this case, the formula in cell C54 returns the array {65, 0}, and therefore the cell C54 displays the number 65, and cell D54 displays the number 0 (that is, the second item in the array *spills* into the adjacent cell D54. Similarly, the arrays returned into C55 and C56 spill into the corresponding cells in column D.

Clara can make changes to her budget just once in the sheet-defined function, and they automatically propagate to the three scenarios.

Turning to probabilistic reasoning, we start with the observation that a SIP, the underlying representation for a probabilistic uncertain value, is simply an array. We can achieve the effect of Monte Carlo modelling by mapping the sheet-defined function for our budget over a SIP array. The array is drawn from the distribution modelling our probabilistic uncertainty about the parameter.

Here is how Clara can do this in her situation:

```
B58 = SIP.TRIANGULAR(B54,B56,B55,1000) // column vector of samples
C58 = VMAP(B58,Budget) // returns: SIPs with means {15.51, 25%}
```

The column vector in B58 has 1000 samples from a triangular distribution. The function call VMAP(B58,Budget) calculates $\{b_i, o_i\} = Budget(s_i)$ for each sample s_i with $i \in 1..1000$, and returns a $[1000 \times 2]$ array $\{b_i, o_i \mid i \in 1..1000\}$. In this example, there is only a single uncertain parameter. To handle N uncertain inputs, the sheet-defined function would take N arguments, and the input would be an N-column array of random samples.

We are simply using the advanced encapsulation afforded by sheet-defined functions to more robustly implement the Monte Carlo simulation idiom described earlier in Section 1.3! This is a more robust implementation as the logic for each trial is defined exactly once and can be easily modified.

Sheet-defined functions and arrays are general-purpose tools, and so there is no inbuilt interface for viewing the resulting array as a probability distribution. Still, it is possible to write formulas, or even sheet-defined functions, to calculate summary statistics of the resulting array.

[7] `https://support.office.com/en-us/article/Dynamic-arrays-and-spilled-array-behavior-205c6b06-03ba-4151-89a1-87a7eb36e531`

3 End-user Behaviour with Uncertainty

Clara's fictitious example is useful to illustrate the differences between formalisms, but is clearly contrived. In order to ground our analysis in examples of real user behaviour, we designed and conducted an interview study.

We aimed to investigate the following questions:

1. What types of uncertainty do spreadsheet users deal with?
2. How do they manage these various types of uncertainty?

We conducted semi-structured interviews of 11 participants, who walked us through their spreadsheets. We analysed the audio transcripts of these interviews, identifying six qualitatively different types of uncertainty, as well as six categories of strategies participants used to cope with the uncertainty. A summary of our study protocol, and the results, follows. These results are presented in greater detail in our paper "Somewhere Around That Number"[3].

3.1 Interview Study

Participants We interviewed 11 participants who worked in finance, construction, IT consulting, the oil and gas industry, business administration, and academic research. The size of participants' spreadsheets ranged from 40 rows to thousands of rows. Participants were recruited using convenience sampling via email invitation and were eligible to participate if they used spreadsheets that contained uncertain data. We did not filter participants based on whether they used spreadsheets for work or personal use, but all participants we recruited dealt with uncertainty in spreadsheets for work purposes. To elicit participation, the invitation gave several examples of spreadsheet tasks which could involve uncertainty, such as budgeting, planning, business forecasting, data collection and analysis, scientific modelling, and making predictions. Interviews lasted on average 60 minutes, and participants were reimbursed with a £30 voucher for an online store.

Procedure We asked participants to bring one or more of their own spreadsheets which contained uncertain data to the interview session. They were instructed to remove any sensitive information that they did not want to share.

In the first part of the session, participants were asked to talk about their work, and how uncertainty and spreadsheets are a part of this work. In the second part of the session, we discussed if participants gain insights from uncertainty, what tools or strategies they use to gain this insight, and what challenges they perceive in doing so. In the final part, we asked participants to walk us through their spreadsheets, and explain how these spreadsheets were constructed, and what they were used for. The session was audio recorded, and participants' walk-through of their spreadsheets was screen recorded.

Data analysis The audio recordings were transcribed verbatim and analysed using iterative coding based on an inductive approach of thematic analysis [5]. There was no pre-existing coding scheme, but we did approach the data with a specific focus to uncover uncertainty types, and user strategies for managing uncertainty. Through a detailed analysis of the transcripts, we identified key features of the spreadsheets and work practices that related to participant concerns with uncertainty.

3.2 Findings

Types of uncertainty Based on participants' descriptions of their spreadsheets during the interviews, we identified six types of uncertainty in spreadsheets: *estimates, dynamic data, errors, missing data, unfindable data,* and *untraceable data*.

1. *Estimates* were the most common type of uncertainty among participants, and refer to approximated values of which the precise value is not known, such as the expected profit of a project: *'We're talking about the future. We don't know exactly what's going to happen. All we can do is make best estimates'* (P8).
2. *Dynamic data* refers to data of which the values changed dependent on time, for instance stock market information.
3. *Errors* were either (1) data that users believed to be incorrect based on their prior knowledge and expectations (these could be caused, for example, by measurement errors, transcription errors), or (2) a spreadsheet error value, such as those created by mis-typed formulas, or formulas receiving arguments of incorrect types, or broken links to external sources. Spreadsheet errors and their sources are an important area of research in their own right, with sophisticated existing taxonomies [19] that we shall not replicate here.
4. *Missing data* were values that were not recorded in the dataset, such as gaps in measured sensor data.
5. *Unfindable data* is information which in principle could be computed from data contained within the spreadsheet, but was hard to extract, and was thus experienced by the user as uncertain. If users were unable to find the information, they regularly used their own estimate instead. For example, P7 dealt with timesheets which gave an overview of hours that all employees of his department had worked per day. He wanted to know how many of these hours were worked on the weekend, but did not know the correct spreadsheet formula to retrieve this data from the timesheet: *'There's a second unknown, which is the weekends [...] Well for me it's difficult, I'm sure there's probably people that can extract it out there'* (P7). This type of uncertainty is particularly interesting, because its presence depends on the circumstances of the spreadsheet user; it is not merely a static property of the data itself.
6. *Untraceable data* refers to data for which the source could not be traced. For instance, participants described situations where it was unclear whether data

they received from other people was derived from a computational model, or whether it was *'completely based on their [colleagues'] intuition'* (P11).

Our claim is that these six categories are useful distinctions between different kinds of unknowns arising in our sample of spreadsheets, and by inference in the whole population of spreadsheets. Estimates, Errors and Missing data are common types of uncertainty found in earlier work [4,33]. In addition to these uncertainty types, we found additional three types of uncertainty: Dynamic data, Unfindable data, and Untraceable data.

A further distinction from the literature on uncertainty is between *aleatoric uncertainty*, due to some random process, and *epistemic uncertainty*, due to lack of knowledge. Our classes of estimates and dynamic data are aleatoric uncertainties, while our classes of errors, missing data, unfindable data, and untraceable data are epistemic uncertainties.

Participants' strategies to cope with uncertainty Participants described various strategies to cope with uncertainty in their spreadsheets. Through our analysis we identified 35 different strategies, which were categorised into six high-level categories: *Minimise, Understand, Communicate, Ignore, Exploit,* and *Add* uncertainty. This is an extension of the categories observed by Boukhelifa et al. [4], with a specific focus on spreadsheets (the original categorisation was tool-agnostic).

1. *Minimising* uncertainty was the most common strategy. Examples of Minimise strategies were to acquire more data, or compare the data with historic data from past situations to try and come to more accurate estimates.
2. The second most common strategy was to *Understand* uncertainty. Participants discussed uncertain data with colleagues, read literature on the subject, researched why the dataset was uncertain, and compared a subset of alternative scenarios.
3. Participants also tried to *Communicate* uncertainty to both themselves and to others. This type of strategy did not necessarily improve people's understanding of why data was uncertain, but its aim was to highlight that there was uncertainty present in the data. To communicate uncertainty to themselves, participants gave spreadsheet cells different colours, or added comments. Participants communicated uncertainty to others through presentations, reports or by providing a verbal narrative.
4. Three participants said they also *Ignored* uncertainty at times. P6, P10 and P11 explained it could be difficult to conduct analysis on a dataset that contained errors or missing values, and these were removed during the analysis process.
5. Sometimes however, it was valuable to know there was uncertainty in a dataset. In this case, an *Exploit* strategy was used, and the uncertainty was extracted or quantified. For example, P11 exploited missing values by adding weights to them in model building, based on how often they occurred. The amount of missing values in a dataset could provide valuable information about what caused the uncertainty, and how important they were to consider.

Type of uncertainty	**Definition**	**Example quote**
Estimates	Data approximated by the user of which the precise value is not known, such as the expected number of attendees to an event.	*'The final value will be somewhere around that figure. However, I haven't gotten to the point yet where I'm able to say: it's in this range, it's in the +/- 3% range'* (P8).
Dynamic data	Data which is not static and changes over time, such as stock market information.	*'When you go to open that [spreadsheet] next month, the information has changed'* (P9).
Errors	Data that contains errors, such as formula or transcription errors.	*'You also get a few very unusual values, I seem to get negative infinities quite a lot. Which clearly is not possible (...) So there's a whole load of error values in there'* (P11).
Missing data	Data that is missing from the data set, such as gaps in measured sensor data.	*'The weather file that's used to generate the data there in the other program is missing large chunks of wind data (...) That was one of those kind of reports that was fairly heavily caveated as being 'We've had to make quite a lot of numbers up here, to get any idea of what might happen with this'* (P6).
Unfindable data	Data which technically is contained within a spreadsheet, but which cannot easily be found by the user, such as the total amount of hours worked on the weekend in an employee timesheet. Usually, being unable to find it, the user uses an estimate instead.	*'There's a second unknown [in the spreadsheet], which is the weekends. (...) I think that kind of formula that we're trying Excel to do is probably difficult. Well for me it's difficult, I'm sure there's probably people that can extract it out there'* (P7).
Untraceable data	Data from which the user cannot trace its original source, and whether or how it is calculated, such as subjective estimates made by other people, or complex and inaccessible formulas.	*'Sometimes it would just be numbers in a spreadsheet, and you couldn't really identify what the hell was going on (...) And half the time, to actually check it, the easiest way is almost to make a new spreadsheet doing it your own way, and see if you get the same number'* (P6).

Table 2: An overview of each of the six types of uncertainty, showing the definition of each, and an example quote where a participant described the type of uncertainty.

Coping strategy	**Definition**	**Example quote**
Add	The user adds additional uncertainty to a spreadsheet. For example, the user is unable to find data in a sheet, and uses an estimate or interval instead. In this situation, data is contained within a spreadsheet, but cannot easily be found by the user, and uncertainty is added by using an approximated value. Alternatively, a collection of data points is reduced to one data point, such as the average value, to make a large spreadsheet easier to view and digest.	*'A lot of the time, there are those things which just aren't [worth the effort to figure out the right formula to extract data]. So they either get done quite slowly, the manual way, or they don't get done at all. And you get people putting like fudge figures (...) and say, 'Oh well it's anywhere between here and here. That's the best we can do'* (P6). *'It gets very confusing when I start having dozen sheets, with 1,000 columns, and 10,000 rows in each. (...) I have a range of values, but the algorithms only give me an average, across say 100 points'* (P11).
Communicate	The user communicates uncertainty to others verbally, and through reports and presentations. Users also communicate uncertainty to themselves, by highlighting cells in their spreadsheets that contain uncertain data.	*'Although we provide estimates, I do provide an estimate which is a hard figure. But what I tell them [clients] is: it's around that figure'* (P8). *'Quite a lot of colouring. Just to highlight particular aspects. So I'd do green for a particular area of what I think 'These are definites'. And light-blue or something for unknowns'* (P4).
Exploit	The user uses the amount of uncertainty in a spreadsheet as a valuable piece of information about the data.	*'Sometimes it [uncertainty] contributes to the forecast, because you want to know sometimes if there's a specific reason for the missing data'* (P10).
Ignore	The user ignores uncertainty, by removing it from the spreadsheet or replacing it with other values.	*'I use a filter on Excel to filter the values (...) So I try to identify them and then find and replace with NAs, most of the time'* (P11).
Minimise	The user minimises uncertainty, by acquiring more data, or discussing it with colleagues.	*'We will liaise with our front office to say, 'Does this look correct? Have things like this happened in the past?' If it's chartered territory, the stuff that we've seen something like this before, we can get more of a better estimate'* (P3).
Understand	The user tries to understand uncertainty by reading literature, discussing it with colleagues, plotting data, evaluating the data source, and analysing a subset of possible scenarios.	*'I would read as much around the literature as I possibly can, I will get in different views that people have (...). And then I'll work from all that to try and inform myself'* (P2).

Table 3: An overview of the six categories of coping strategies, its definition and an example quote where a participant gave an example of the strategy.

6. Interestingly, one coping strategy to deal with uncertainty was to minimise one type of uncertainty, by *Adding* another type of uncertainty. For example, if participants dealt with unfindable data that they could not extract from a spreadsheet, they used an estimate instead. Uncertainty was also added by only considering a summary or a subset of the data. P11 dealt with datasets of measured sensor data, which could be tens of thousands of rows. To be able to view and easily digest this data, he would replace multiple data points with one data point, such as the average value of those data points.

	Dynamic data (52, 15%)	**Estimates** (175, 50%)	**Errors** (47, 13%)	**Missing data** (44, 12%)	**Unfindable data** (24, 7%)	**Untraceable data** (10, 3%)
Qualitative	Some	Good*	Good*	Weak	None	None
Possibilistic	Some*	Good*	None	Some	None	None
Probabilistic	Good*	Weak	Some*	Some*	None	None

Table 4: An overview of the six types of uncertainty (the counts and percentages indicate the frequency of each uncertainty type in our dataset of 352 observations) and the three formalisms. The table indicates how well we think a type of uncertainty could have been supported by a particular formalism, on the scale: none, weak support, some support, good support. We used the following rough guidelines to make this classification: None: this formalism will not help the users in this uncertainty type. Weak: this formalism will help with this uncertainty type, but has significant limitations in comparison to the others. Some: this formalism would help in some cases where users encounter this uncertainty type. Good: (features supported by) this formalism would help in many or most cases where users that encounter this uncertainty type. An asterisk (*) indicates that we observed at least one participant directly state that type of uncertainty would be supported by that formalism.

4 Design Implications of the Interview Study

So if the makers of spreadsheet software wished to implement user-facing tools for the representation and management of uncertainty,[8] what should they focus on? What features would solve the most common use cases we observed? In other words, what should we build to get the most bang for our buck? In this section we present some *implications for design* (notwithstanding the criticism of that phrase [34,7]).

[8] We began this paper with a set of premises leading to the hypothesis: introducing probabilistic programming to spreadsheets is a Good Idea. Through our user research, we refined our ideas from focusing on probabilistic programming, to ways of representing and managing uncertainty more generally.

Formalism - Type of uncertainty	**How the formalism would be used**	**Example quote**
Qualitative - Estimates	The user can input their own values of what they think an estimate is.	*'We have to convert that qualitative scenario into some quantitative numbers, to say 'OK, if X wins, it's likely that the market's going to react a little bit better, just because of her economic policies, therefore we would expect GBP to strengthen.' And then we would revise our forecast based on that, going forward. But then again, those are unknown numbers, we're just using, we would use estimates to try and say, 'This is where we'd end up'* (P3).
Possibilistic - Dynamic data	The user can input intervals to indicate the range that dynamic data can fluctuate in, and/or can compare a subset of outcomes of dynamic data.	*'You'd have a day rate, and multiply it in terms of days. And then you add a certain percentage on afterwards, just so you have some space'* (P4).
Possibilistic - Estimates	The user can compare a subset of possible estimates.	*'You get a huge extra benefit from looking at one policy to looking at two policies. It's a bit more time-consuming, it takes twice as long to run, but it's worth doing. But to go from two to three policies, it's not so clear that you get enough extra benefit from that, from the extra complexity'* (P2).
Probabilistic - Dynamic data	Based on historic values, the user can quantify the likelihood of future values.	*'Once you get the data, then it becomes a modelling exercise. Like what is my chance of selling something next month? Well that depends on how much of that thing has been selling this month'* (P9).
Probabilistic - Errors	If the data errors in a spreadsheet are quantifiable, probabilistic measures can be used to track and quantify uncertainty as it propagates through the calculations of a spreadsheet.	*'At every stage, a lot of the time we have: if we know we've got however many unknown values in spreadsheet 1, by the time we get to spreadsheet 10, there's a debate about whether the data's of any use at all, because the errors propagate, and multiply. Now I don't have a measure of how that affects it. I'd love to be able to measure that, but I have no idea how to do that within the tools'* (P11).
Probabilistic - Missing data	The user can make estimates regarding missing data by analysing the distributions of the data that does exist in the spreadsheet.	*'If we get certain data, and we want to make sure: does that follow a certain distribution, or something like that? Or is that data based on a certain distribution? There are tests that give you a p value, very likely that this data point's origins are the same as the other data points. (...) So that gives you some confidence data in how well you can rely on this model, on that model, and stuff like that'* (P10).

Table 5: Participant quotes exemplifying how types of uncertainty could have been supported by a particular formalism.

In the previous section we have been introduced to a broad categorisation of uncertainty formalisms into Qualitative, Possibilistic, and Probabilistic, and how the formalism determines the family of features that can be implemented to support them – tags for Qualitative uncertainty, the scenario manager or intervals for Possibilistic uncertainty, and so on. At the same time, our research has identified six types of uncertainty in spreadsheets. We therefore examine how each formalism can support the management of each type of uncertainty, on a case by case basis, by considering how the user might use the various implementations of each formalism in those cases.

1. *Dynamic data* changes over time, either refreshing automatically or needing manual re-entry. A Qualitative tag may help to indicate the presence of uncertainty. A Possibilistic or Probabilistic representation may help to keep a record of historic values for a cell, with a Probabilistic representation capturing additional information, helpful for statistical inference.
2. *Estimates* can be supported by Qualitative tags, which can propagate through the calculation dependency chain so that it is made apparent which calculations depend on estimated data. In entering estimates, users may also be able to provide upper and lower bounds, or a set of possible values, which are well-supported by the Possibilistic formalism. Finally, if possible, it would be ideal to elicit a formal probability distribution, but our experience with users suggests that non-expert end-users find it challenging to understand and confidently assign parameters to a probability distribution. Consider the difference between a non-expert having to produce "upper and lower bounds" for a cost in a budget, and having to produce a "mean and standard deviation for a Gaussian distribution". This makes the Probabilistic scenario a weak overall fit for estimates, although we recognise that it might fit the needs of some experts very well.
3. *Errors* benefit most from Qualitative tagging, as a way of annotating their presence and communicating them to others. It is unclear whether Possibilistic or Probabilistic uncertainty can help in error cases, although some cases of data entry errors can be detected using the insight that an erroneous data value is often very unlikely given the distribution of the rest of the data.
4. *Missing data* can be supported by Qualitative tagging, if calculations are done on ranges with missing values. Missing values can be inferred and represented either in a Possibilistic manner (for instance, if the missing value is known to be a member of a finite set) or in a Probabilistic manner (for instance, if the missing value is part of a dataset with a known or calculable distribution).
5. *Unfindable data* – that is, answers that are prohibitively difficult to extract from the data in the spreadsheet, even if it is possible in principle. This type of uncertainty cannot be solved by implementing better tools for representing and manipulating uncertainty; the solutions here depend more on enabling users to store and retrieve data in different ways, from enabling the pivoting and refactoring of existing spreadsheet layouts, to better assistance in formula authoring.

6. *Untraceable data* might be supported using a Qualitative tag, to indicate and communicate its presence. However, better tracking of data provenance would help deal with the issue more holistically. Users should be able to inspect the history of edits to a spreadsheet cell, and answer the questions: who edited this cell and when? Was the data manually typed in, copied in from a document, or written by a macro or other script? The implementation of some of these ideas (such as tracking and linking to documents from which data is copied) might go beyond the scope of the spreadsheet software, and require the participation of other tools and the operating system to track provenance.

In summary, solutions focused around enabling users to formally *represent* uncertainty offer varying levels of support dynamic data, errors, estimates and missing data, but are unlikely to help with unfindable data and untraceable data. Table 4 shows a matrix of the three categories and the six types of uncertainty, summarising the discussion above. For many pairings where we claim that the formalism would support the user, we provide a participant quote in Table 5 – these entries are indicated with an asterisk (*).

It should be noted that the user behaviour we observed was shaped and constrained by the affordances of spreadsheets today; users may appear to make limited use of uncertainty in spreadsheets simply because of the lack of native support for uncertainty in spreadsheets. There may be a larger latent demand for such features, which is currently unobservable due to this 'technological determinism'. Or, there may be no such demand and the observed behaviour is a true reflection of society's needs. It is impossible to tell using our interview methodology alone.

Bearing this in mind, let us return to the question of what designers of spreadsheet systems can take away from this. One clear conclusion is that no single formalism can solve most user problems. Of Qualitative, Possibilistic, and Probabilistic formalisms, Qualitative tagging appears to be the most widely applicable, but what the user can do with it is limited. Our recommendation is therefore to offer features from multiple formalisms. Furthermore, if we were to pick one to exclude, it would be Possibilistic, as the technique of managing different scenarios across different columns (or rows, or cells, or sheets, or even whole workbooks), as well as tinkering with values in cells, are both already effective coping strategies in many use cases for Possibilistic data. This recommendation comes despite the fact that Possibilistic data is likely to be easier to input and reason about than Probabilistic data, for most end-users. Finally, solutions focused around *representing* uncertainty are unlikely to help with unfindable and untraceable data—for those cases, we need tools offering education, assistance with authoring, layout and refactoring, and provenance tracking.

5 Conclusions

In this paper, we formally acknowledge the human activity of *end-user probabilistic programming*, in which a computer user, likely not a statistical expert, creates

models to support their own decision making. While probabilistic programming is a powerful tool, its widespread adoption is limited by the high statistical and programming expertise required. Spreadsheets, on the other hand, enjoy the status of being the *premier* end-user programming tool, and are arguably the venue of much of the world's data and decision making. What if we were to bring probabilistic programming to spreadsheets?

As a stepping stone, we consider the simpler question of bringing native support for *uncertain* values to spreadsheets. The choice of whether we support qualitative (such as simple tags), possibilistic (such as sets or intervals) or probabilistic (such as distributions) uncertainty will have profound implications for the user experience, which we illustrated through the fictitious case study of Clara's budget. To study these implications empirically, we interviewed 11 spreadsheet users, focusing on their current use cases and coping strategies for uncertainty.

We produced a taxonomy of uncertain data as experienced by end-users, and analysed how each uncertainty formalism could support the different categories, finding the following:

- No single formalism emerges the clear winner, and though Possibilistic uncertainty is well-understood by end-users, the data suggests that it would add the least benefits, since end-users already have several effective strategies for coping with it.
- We also identified that users experience types of uncertainty, such as unfindable and untraceable data, that cannot be solved by representations of uncertainty alone.
- Estimates were the most common type of uncertainty experienced by end-users in our sample, with 157 occurrences (nearly 50% of all observations). While we should be careful with drawing quantitative inferences from small samples such as ours, this is still an indication that Estimates are an important category of uncertainty cases.

As well as considering special-purpose uncertain values, we showed idioms for representing probabilistic and possibilistic uncertainty using the general-purpose concept of sheet-defined functions.

The idea of probabilistic programming in spreadsheets is certainly alluring. But for full effect, we must ensure that the idea works for the tens (or even hundreds) of millions of spreadsheet users facing uncertainty in their decision making. At present, neither uncertain values nor sheet-defined functions are available in mainstream spreadsheets. Will one or both revolutionise probabilistic programming in spreadsheets? Time will tell.

Acknowledgements Thanks to Breck Baldwin and Matthijs Vákár for information regarding Stan. We are grateful to Alan Blackwell, Eunice Jun, Tom Minka, Simon Peyton Jones for their helpful comments on a draft of this paper.

References

1. Benton, N., Hughes, J., Moggi, E.: Monads and effects. In: APPSEM. Lecture Notes in Computer Science, vol. 2395, pp. 42–122. Springer (2000)
2. Blackwell, A.F., Burnett, M.M., Peyton Jones, S.L.: Champagne prototyping: A research technique for early evaluation of complex end-user programming systems. In: VL/HCC. pp. 47–54. IEEE Computer Society (2004)
3. Borghouts, J., Gordon, A.D., Sarkar, A., O'Hara, K.P., Toronto, N.: Somewhere around that number: An interview study of how spreadsheet users manage uncertainty. arXiv preprint arXiv:1905.13072 (2019)
4. Boukhelifa, N., Perrin, M.E., Huron, S., Eagan, J.: How data workers cope with uncertainty: A task characterisation study. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 3645–3656. ACM (2017)
5. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative research in psychology **3**(2), 77–101 (2006)
6. Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., Riddell, A.: Stan: A probabilistic programming language. Journal of Statistical Software **76**(1) (2017)
7. Dourish, P.: Implications for design. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. pp. 541–550. ACM (2006)
8. Gelman, A., Hill, J.: Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press (2007)
9. Gilks, W.R., Thomas, A., Spiegelhalter, D.J.: A language and program for complex Bayesian modelling. The Statistician **43**, 169–178 (1994)
10. Goodman, N., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In: Uncertainty in Artificial Intelligence (UAI'08). pp. 220–229. AUAI Press (2008)
11. Gordon, A.D., Graepel, T., Rolland, N., Russo, C.V., Borgström, J., Guiver, J.: Tabular: a schema-driven probabilistic programming language. In: Principles of Programming Languages (POPL'14) (2014)
12. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Herbsleb, J.D., Dwyer, M.B. (eds.) Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014. pp. 167–181. ACM (2014)
13. Gorinova, M.I., Sarkar, A., Blackwell, A.F., Syme, D.: A live, multiple-representation probabilistic programming environment for novices. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 2533–2537. ACM (2016)
14. Goudie, R., Thomas, A.: MultiBUGS: A parallel implementation of the bugs modelling framework for faster bayesian inference (2019), talk at workshop on Advances and challenges in Machine Learning Languages, `https://people.ds.cam.ac.uk/rg447/2019-05-21-goudie-acmll-slides.pdf`.
15. Grad, B.: The creation and the demise of VisiCalc. IEEE Annals of the History of Computing **29**(3), 20–31 (2007)
16. Hermans, F., Jansen, B., Roy, S., Aivaloglou, E., Swidan, A., Hoepelman, D.: Spreadsheets are code: An overview of software engineering approaches applied to spreadsheets. In: FOSE@SANER. pp. 56–65. IEEE Computer Society (2016)
17. Hyvönen, E., De Pascale, S.: A new basis for spreadsheet computing: Interval solver for Microsoft Excel. AI Magazine **21**(4) (2000)

18. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., et al.: The state of the art in end-user software engineering. ACM Computing Surveys (CSUR) **43**(3), 21 (2011)
19. Kulesz, D., Wagner, S.: Asheetoxy: A taxonomy for classifying negative spreadsheet-related phenomena. arXiv preprint arXiv:1808.10231 (2018)
20. Lunn, D., Jackson, C., Best, N., Thomas, A., Spiegelhalter, D.: The BUGS Book. CRC Press (2013)
21. Martin, O.: Bayesian Analysis with Python: Introduction to statistical modeling and probabilistic programming using PyMC3 and ArviZ. Second edn. (2018)
22. McCutchen, M., Borghouts, J., Gordon, A.D., Jones, S.P., Sarkar, A.: Elastic sheet-defined functions: Generalising spreadsheet functions to variable-size input arrays (2019), in submission
23. Minka, T., Winn, J., Guiver, J., Zaykov, Y., Fabian, D., Bronskill, J.: Infer.NET 0.3 (2018), Microsoft Research Cambridge `https://dotnet.github.io/infer`
24. Nardi, B.A., Miller, J.R.: The spreadsheet interface: A basis for end user programming. In: Diaper, D., Gilmore, D.J., Cockton, G., Shackel, B. (eds.) Human-Computer Interaction, INTERACT '90, Proceedings of the IFIP TC13 Third Interantional Conference on Human-Computer Interaction, Cambridge, UK, 27-31 August, 1990. pp. 977–983. North-Holland (1990)
25. Ntzoufras, I.: Bayesian Modeling Using WinBUGS. Wiley (2009)
26. Peyton Jones, S.L., Blackwell, A.F., Burnett, M.M.: A user-centred approach to functions in excel. In: International Conference on Functional Programming. pp. 165–176 (2003)
27. Pfeffer, A.: Figaro: An object-oriented probabilistic programming language. Tech. rep., Charles River Analytics (2009)
28. Powell, S.G., Baker, K.R.: The art of modeling with spreadsheets. John Wiley & Sons, Inc. (2003)
29. Sarkar, A., Gordon, A.D., Peyton Jones, S., Toronto, N.: Calculation view: multiple-representation editing in spreadsheets. In: VL/HCC. pp. 85–93. IEEE Computer Society (2018)
30. Savage, S., Scholtes, S., Zweidler, D.: Probability management. OR/MS Today (Feb 2006)
31. Savage, S.L.: The Flaw of Averages. Wiley (2009)
32. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05). pp. 207–214. IEEE (2005)
33. Schunn, C.D., Trafton, J.G.: The psychology of uncertainty in scientific data analysis. In: Handbook of the Psychology of Science. Springer-Verlag (2012)
34. Stolterman, E.: The nature of design practice and implications for interaction design research. International Journal of Design **2**(1) (2008)
35. Streit, A.: Encapsulation and abstraction for modeling and visualizing information uncertainty. Ph.D. thesis, Queensland University of Technology (2008)
36. Streit, A., Pham, B., Brown, R.: A spreadsheet approach to facilitate visualization of uncertainty in information. IEEE Trans. Vis. Comput. Graph. **14**(1), 61–72 (2008). https://doi.org/10.1109/TVCG.2007.70426, `https://doi.org/10.1109/TVCG.2007.70426`
37. Tolpin, D., van de Meent, J., Wood, F.: Probabilistic programming in Anglican. In: Bifet, A., May, M., Zadrozny, B., Gavaldà, R., Pedreschi, D., Bonchi, F., Cardoso, J.S., Spiliopoulou, M. (eds.) Proceedings of ECML PKDD 2015, Part III. LNCS, vol. 9286, pp. 308–311. Springer (2015)

38. Winston, W.L.: Microsoft Excel 2019 Data Analysis and Business Modeling. Microsoft Press, sixth edn. (2019)
39. Wu, M., Perov, Y.N., Wood, F.D., Yang, H.: Spreadsheet probabilistic programming. CoRR **abs/1606.04216** (2016), (see also the Scenarios tool at `invrea.com`)